EONS: Efficient Execution of Large Workloads on Elastic Heterogeneous Resources

Carsten Griwodz, **Pål Halvorsen** (Simula Research Laboratory), **Dag Johansen** (University of Tromsø)

FRINATEK/FRIPRO Project Proposal

1 Relevance relative to the call for proposals

In the current and future industry and society, there will be an increasing number of systems storing and processing large amounts of data. This is the next frontier for innovation, competition and productivity [20] where there are currently large initiatives both in the EU [1, 2] and the US [3, 4, 5]. Example data sets stem from areas like medicine, meteorology, genomics, connectomics, physics, biology, environmental research, Internet search, sports, finance and governmental informatics. Here, there are massive computational demands, but also huge demands for I/O and communication and for timeliness and deadlines. Additionally, there are often dependencies between processing steps. In a large computing cluster like a grid or a cloud, an important challenge is thus to execute the many concurrent computations in an efficient and dynamic manner where available resources, processing, communication, dependencies and timeliness must be taken into account when mapping tasks to processing cores.

As such, the aim of the *EONS* research project is to perform basic research in the area of development of parallel programming and parallel processing in the context of future distributed large-scale heterogeneous systems. We aim to research and develop *concepts and mechanisms* that will enable the development of *software* for these next-generation *big-data* applications. This is achieved by solving fundamental challenges for the dispatching, division, scheduling and identification of tasks that can run correctly in parallel in a shared distributed system of heterogeneous computing resources in complex topologies.

2 Background and status of knowledge

Problem area The number of services requiring processing and delivery of large amounts of data is steadily increasing. In this respect, there is often a dynamic number of input sources and output sinks, and the amount of available processing power is often varying depending on competing workloads. One example, which we functionally have built in the iAD CRI, but not scaled, is shown in figure 1 where we have a large scale sport (soccer) arena system [16] that combines and processes data from motion and medical sensors, expert analytics and video cameras. Then, it delivers statistics, events and personalized highlights to a dynamic number of different users. In this example, we use a (varying) number of HD cameras to capture the game, both statically mounted and potentially using spectator devices (like smart-phones or Google-glasses), generating terabytes of video data per hour. Furthermore, the sensor systems deliver millions of data records per hour, and the game experts provide textual descriptions of the game. All the different kinds of input may require individual processing as well as processing of the combined information. The individual users

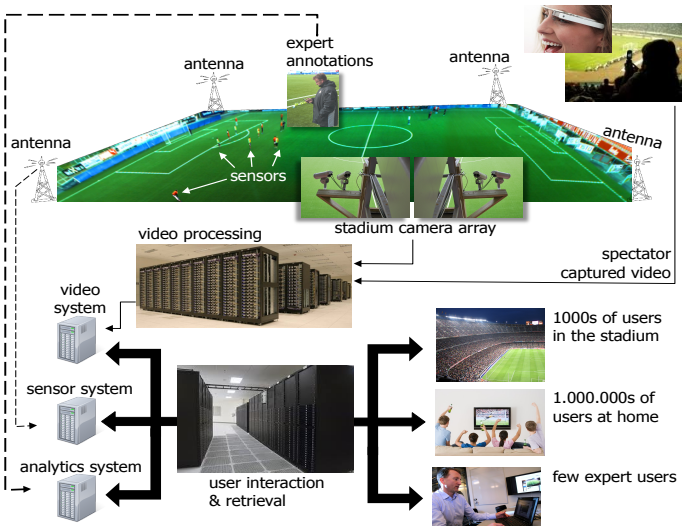
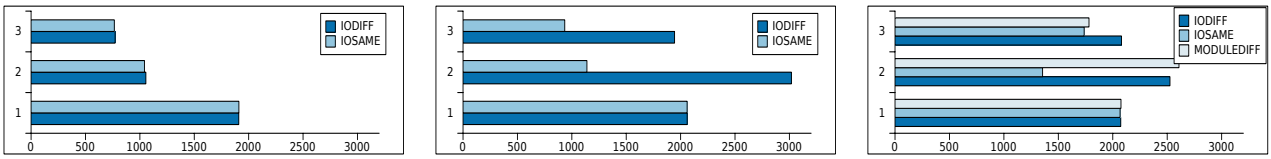


Figure 1: System setup at Alfheim stadium (see [16]).

are then able to extract their own high-
lights on the fly in real-time, game analyses are delivered to the coaches during the game to make better decisions, etc. Similar large scale I/O and processing demanding scenarios are present in the examples listed in section 1, where a major limitation of the current state of the art is the lack of support for 1) efficient execution of such large workloads on elastic heterogeneous resources in general and 2) delivery of results in real-time. To achieve efficient processing and real-time capabilities, data must be processed at all stages of the pipeline using devices ranging from mobile phones to large mainframes and data centers with multiple heterogeneous processing engines, i.e., the current trend where everything is pushed to the cloud does not work alone due to huge data sets, varying resource availability and real-time constraints.

Nevertheless, distributed and parallel processing have been important and popular research areas for a long time, and in a never-ending quest for solving increasingly complex tasks, software developers have consistently pushed the boundaries for computational demand. As the physical limitations of pushing the clock frequency on a single-core CPU became apparent, new hardware solutions evolved. The focus moved to increasing the number of cores. This effectively forced the programmers to parallelize their applications to increase performance.

With parallelisation, modern distributed computing systems often provide the required processing power for large-scale data processing. However, their increasing complexity is a challenge. Existing and future topologies exhibit a range of capabilities where computing nodes consist of multiple heterogeneous processing engines including multi-core CPUs, digital signal processors (DSPs), field-programmable gate array (FPGAs) and graphics processing unit (GPUs). To further complicate the picture, these devices often vary their behavior between generations. See for example the experimental results shown in figure 2 where the interaction between steps in a processing pipeline gives (very unexpected) performance differences based on the number of threads, their affinity and the used microarchitecture [13]. Similar observations can be made on GPUs and FPGAs. Furthermore, these nodes are connected by a variety of networks, from wireless networks to high-speed interconnects. Communicating via a network adds latency and can introduce delays detrimental to the progress of the application. Moreover, parallel applications on shared-memory architectures need to synchronize threads of execution when accessing shared data. Using synchronization primitives correctly requires extensive experience, otherwise one might thrash locks, serialize parallel steps, or create deadlocks, starvation, race conditions, etc. Similar challenges rise on non-shared memory systems, and in addition, I/O and communication between processing steps must be taken into account. It also becomes necessary to distribute data and synchronize state between nodes in the topology. Fault-tolerance is required to handle situations where nodes in the network become unavailable, and these are just a few of the obstacles that must be overcome. Thus, the increased complexity results in a lack of portability of the existing scheduling designs, and moving from a *sequential* mode of operation



(a) Sandy Bridge (b) Nehalem (c) Bulldozer

Figure 2: Execution times (x-axis in ms) of video processing pipeline correcting lens distortion, rotating and cropping the images in a video sequence using different thread affinity strategies (lower is better) with different numbers of threads (y-axis) on different modern microarchitecture. On Intel's Nehalem and Sandy Bridge processors, we tested two different execution modes, i.e., one mode where the processing of a frame was done on the same core that performed the I/O (IOSAME), while the second mode processed the frame on a different core than the I/O thread (IODIFF). Since the AMD Bulldozer architecture has CPU modules, we added an additional mode for this architecture, in which processing threads were executed on different CPU modules than the thread handling I/O (MODULEDIFF). See [13] for more information.

to *parallel* execution is significantly *more demanding* on the developers.

The emergence of all this special hardware provides a potential for large performance increases. However, given the current situation where cores and communication channels have different capabilities and where frameworks like OpenCL and CUDA provide APIs and compilers that attempt to conceal the fact that tasks will be executed on a GPU, the challenge is to efficiently utilize resources. The application developers must have in-depth domain-specific knowledge of algorithmic solutions to a task, the implementation options that allow for parallel implementations, the topology of the available execution environment, and even domain-specific knowledge of the individual architectures available for executing tasks. A large burden is then placed on the developers as they must consider the complexity that arises from the interplay of both application and topology of resources. This gives rise to the necessity of defining a programming model that is capable of defining the interaction patterns between components in a manner that lends itself well to automatic parallelization and distribution, while providing the flexibility to define the interaction patterns of these complex applications in an intuitive manner.

As such, *distributed computing* improves the *scalability* of an application, but introduces *large overheads and complexity* which the developer must consider when partitioning the work. Developing applications that utilize distributed heterogeneous computing resources is therefore a *hard* task, and due to the complexity, traditional parallel algorithms cannot be applied directly. Ensuring that the utilization of the resources is maximized is even *harder*. This means that it is essential to research and develop mechanisms and concepts that make it possible for a programmer to write an application in a manner close to their intrinsic understanding of how to structure a program. Simultaneously, we must ensure that the run-time system executing the application is capable of maximizing the utilization of the available resources by applying efficient scheduling algorithms at different levels.

Related work Many-core systems, such as graphic processor units (GPUs), digital signal processors (DSPs) and large scale distributed systems in general, provide the required processing power for executing computationally intensive applications, but taking advantage of the parallel computational capacity of such hardware is much more complex than single-core solutions. In addition, heterogeneous hardware requires individual adaptation of the code, and often involves domain-specific knowledge. All this places additional burdens on the application developer.

As a consequence, several frameworks have emerged that aim at making distributed application development and processing easier, such as Google's MapReduce [11], IBM's System S [15] and Microsoft's Dryad [18]. These frameworks are limited by their design for batch processing of large amounts of data, with few dependencies across a large cluster of machines. Modifications and enhancements that address bottlenecks [9] together with support for new types of workloads and additional hardware exist [23, 10, 17] together with languages for building more complex pipelines [21, 22]. Despite their potential, these frameworks are limited to certain application domains – *batch processing*. Distributed multimedia applications, such as 3D video applications, have stricter requirements for flexibility, require support for iterations (cycles in the processing pipeline), deadlines and the ability to express arbitrary processing graphs. Thus, traditional batch processing frameworks do not commonly integrate knowledge of deadlines into the run-time itself. While support for iterations exists, it is not considered in the parallelization stage and rather solved by iterative execution of the workload, which might introduce artificial barriers between iterations, potentially slowing down faster workers. There is also frequently limited support for arbitrary processing graphs, sometimes limiting them to directed acyclic graphs or pre-defined stages.

The Sisal language [14] is an example of a language that can solve some of these issues. It does automatic parallelization by introducing an intermediate processing step that outputs a

data-flow graph. This graph can be examined to determine when parallel execution in the code is possible. Sisal, however, is limited to multi-core machines and is optimized for numerical programs. Cray's Chapel [8] language is an example of a framework with a more complete feature set. It is based on a multi-resolution model, where developers can write abstract high-level code initially, and then further refine their code to match the underlying architecture. Multimedia workloads are not directly supported in the Chapel language, so knowledge of deadlines has to be added explicitly.

With respect to scheduling, a large mount of work exists, ranging from device and workload specific schedulers to general purpose operating system schedulers. However, in the area of the processing frameworks listed above, the de facto standard is a variant of *work-stealing scheduling* [7]. The randomized stealing of tasks is cache unfriendly, and Acar et al. [6] therefore suggest a model that prefers stealing tasks where the worker thread has affinity with that task and gains increased performance. Another approach for cluster scheduling is delayed scheduling [28] taking resource sharing between users and data locality into account in MapReduce-like systems, but there are still the challenges related to iterative workloads and deadlines.

Own activities and related projects in the area The project proposers have all been researching and teaching in the parallel and distributed computing area for several decades. Lately, our work in the area has been performed in the context of the Information Access Disruption (iAD, see http://www.iad-center.com) Center for Research-based Innovation (CRI/SFI) where we have performed research in various areas of distributed processing. We focus on *all* layers of the big data analytics stack ranging from low level operating system support to high-level programming and distributed processing frameworks. For example, for better and more efficient processing, we have developed the Vortex operating system [19] (running for example Hadoop). To increase efficiency of MapReduce and provide a more expressive, declarative programming model, we have proposed Oivos [24], and to reduce the added overhead of layering software on top of MapReduce, Cogset [25] changed the processing architecture to further increase performance. With the Nornir run-time system for parallel processing [27], we addressed many of the shortcomings of these (batch) processing frameworks. Nornir is based on the idea of Kahn Process Networks (KPN). Opposed to MapReduce-like approaches, Nornir supports arbitrary processing graphs, and unlike Chapel it has deterministic execution. In this context, we also proposed modifications to work stealing scheduling [26]. KPNs, however, require some unrealistic assumptions (like unlimited queue sizes), and the existing Nornir programming model is much more complex than that of frameworks like MapReduce and Dryad. It demands that the application developer establishes communication channels manually to form the dependency graph. P2G [12] builds on the ideas of Nornir and is designed specifically for developing and processing distributed real-time multimedia data. P2G supports arbitrarily complex dependency graphs with cycles, branches and deadlines, and provides both data- and task-parallelism. However, there are still programming and execution (scheduling) challenges that are not solved in dynamic and heterogeneous environments.

In the context of our ongoing activities and further relevant application scenarios, we are currently in dialog with "Olympiatoppen", the committee for Olympic Games in Oslo 2022 and "Norsk Toppfotball"/Cisco regarding relevant research in the area of real-time sport data collection, processing and delivery, and with the Intervention Centre (Rikshospitalet, Oslo University) and the Faculty of Health Sciences in Tromsø regarding medical scenarios. Finally, we have a strong iAD-cooperation with Microsoft Development Center Norway (MDCN) and their main big-data analytics team. These activities may also be the foundation for new applications in the intersection between computer science and medicine.

3 Approaches, hypotheses and methods

The *EONS* project will research "system support" for future large scale interactive distributed applications with large requirements for both I/O and processing power[1]. This means that we aim for a more efficient development and execution system in order to ease distributed parallel application development and improve resource utilization in distributed heterogeneous environments, respectively. Thus, the following are important areas to pursue (which are further described in the work packages in section 4.1):

- Formalization of a high level parallel programming model and programming language. There exists several approaches to specify potential parallelism, but for workloads with processing and/or time dependencies, we need to add notions of deadlines and execution orders.
- Compiler and multi-core run-time system. Many run-time systems have been built and are in use, but there are large potentials for more efficient execution and run-time support for the dependencies must be added. Scheduling and mapping of tasks to processing engines will here be important.
- Distributed implementation and high-level scheduler optimization. Adding support for multiple machines makes the previous item more complex. The heterogeneity and complexity increase and the communication costs vary more. A high-level scheduler therefore must take this into account, in addition to the competition for resources from different concurrent workloads.

All project participants have experience in the area of experimental research, and the *EONS* project will therefore use evolutionary prototyping to accomplish its goal of investigating and implementing a mechanisms that enable the development and execution of complex, time-dependent and computationally intensive applications. Using simulations to prove the validity of our high and low-level schedulers, scalability, etc., would be possible, but past experience has shown that we are frequently unable to model the complexity of the system correctly. Furthermore, the best venues for publishing research results in this area today require test results from real systems. As such, developing a proof-of-concept prototype running real applications is the only feasible approach for solving issues related to this project.

4 Project Plan

4.1 Structure and work packages

To address the research challenges in the areas listed in section 3, *EONS* is structured into 5 major work packages (WPs) each containing a set of tasks. We expect that the post doc and PhD students are available at project start; this is a feasible assumption because qualified personnel are available both at Simula and UiT. Each work package includes involvement from all project partners, for details on the these partners see section 5.2. The approximate time-plan with the milestones defined in the online form is shown in figure 3.

WP1: Analysis of applications and hardware. As stated before, we have worked in this area for several decades, and we do have knowledge of application requirements and hardware capabilities. However, both these areas move forward fast, and we must therefore continuously

[1] Researchers have worked on *similar* problem areas for a long time under many different names such as grid-, high performance-, cluster-, scientific- and cloud computing. There are many similarities between these areas like large data sets, distributed resources, etc., and they still share many of the same challenges like efficient resource utilization. Additionally, new big-data challenges often include interactivity (response time, deadlines), multi-user resource sharing and how the applications are used (interactivity).

monitor the development both on the application and the machine resource sides. In this respect, we do have research group members that have strong contacts to hardware manufacturers like Dolphin, Numascale, Intel and NVIDIA, and we have ongoing deep collaboration with large "big-data" companies (like Microsoft and Accenture) in the iAD project.

Tasks	**T1.1** Extract high level requirements for existing and future applications. **T1.2** Consider implications of supporting emerging and future architectures.
Expected results	Detailed knowledge of existing systems and future trends in both application and hardware development which will be used as input for the other WPs wrt. requirements and capabilities, respectively.

WP2: A high level parallel programming model and programming language. Parallel and distributed programming is hard, and many tools, frameworks and programming language constructs exist to specify parallelism. However, added features like timeliness and processing dependencies and the added complexity of modern heterogeneous microarchitectures make the development a tedious task. The programming model developed in *EONS* will be designed to exploit both task and data parallelism to ensure maximum exploitation of the parallelism in an application. Minimizing the strain on the developer will be accomplished by utilizing structures and ideas that lie close to a programmers intrinsic understanding of program structure. The programming language, and corresponding syntax, expose the programming model to the developer. It is not a goal of this research project to develop a completely new programming language, but our past experience with Nornir, Oivos, Cogset and P2G has taught us that it is necessary to provide a high level interface for the developer to work with. This is essential because it helps ensure adaptation, and greatly aids in the development process when writing applications.

Tasks	**T2.1** Define a generalized programming model **T2.2** Ensure programming model supports intrinsic understanding for programmer **T2.3** Ensure programming model supports parallelization of both tasks and data **T2.4** Define programming language syntax and semantics
Expected results	A development concepts where the programmer easier can develop parallel and distributed applications where timeliness and dependencies can easily be specified. The model should transparently determine the level of parallelism. We expect to be able to publish several papers in parallel programming conferences.

WP3: Compiler and multi-core run-time system. Based on the requirements and the programming model, and the previous models we have developed in Oivos, Cogset, Nornir and P2G, this WP will develop a fully functioning compiler and run-time for multi-core machines. The compiler will generate an abstract syntax tree (AST) from a program written using the high-level programming language developed in WP2. This AST will be used to generate code compatible with a given run-time. The initial target is a multi-core x86 run-time (which by itself behaves very differently on different architectures, figure 2) This run-time should be capable of executing an application, and as such, will be responsible for assigning tasks to the available local processing cores on the machine it is running on. In this WP, it is not expected that distribution to multiple machines, connected by a network, is supported, but it should be capable of maximizing utilization on (homogeneous) multiple cores.

Tasks	**T3.1** Implementation of generalized compiler, with parser outputting AST **T3.2** Implementation of run-time system for an x86 multi-core machine **T3.3** Implementation of simple low-level scheduler **T3.4** Code-generator for generating code that integrates with the x86 run-time[†]
Expected results	A prototype of a compiler and run-time will be developed. The input should be a program developed using the model from WP2, which then will be compiled and executed parallel in a transparent way. We expect to be able to publish several papers in parallel computing, distributed systems and operating systems conferences and journals.

[†]Relying on a native compiler as its back-end

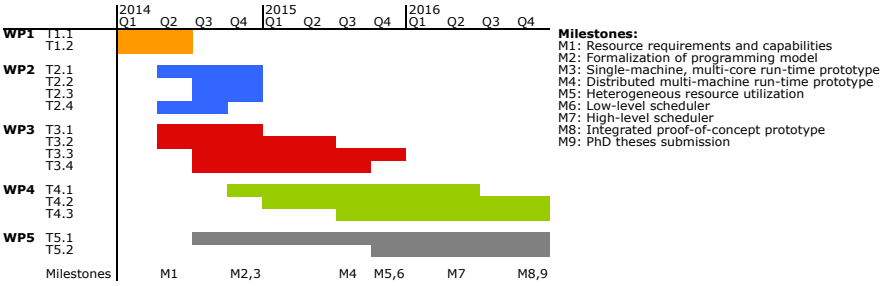


Figure 3: Approximately time-plan with milestones.

WP4: Distribution and high-level scheduler optimization. The performance of a single machine is limited, of course, and large scale processing frameworks (like MapReduce, System S, Dryad, etc.) rely therefore on the use of multiple machines. In parallel with the single machine runtime prototyping, means for distribution will be researched. This work will consist of the distribution and coordination of tasks and the run-time system will thus be extended with communication primitives and capabilities to accept new tasks, gather the required data and correspondingly distribute data to other nodes. This distributed version will be designed to adapt dynamically to the amount of resources available. A major challenge lies in the creation of an efficient high-level scheduler. This scheduler will be responsible for assigning tasks in an efficient manner, reducing communication overhead and maximizing throughput, and distributing load evenly. It will also support co-scheduling of multiple workloads.

Tasks	**T4.1** Extend run-time system with communication primitives **T4.2** Implement high-level scheduler for distribution and coordination of tasks **T4.3** Support elastic resources, i.e., hardware can be added and removed dynamically
Expected results	A proof-of-concept distribution and communication prototype linking and extending the run-time from WP 3 will be developed. Thus, it should be application transparent if a task is executed locally or remotely where the scheduler efficiently maps tasks to processing cores based on current load, processing and latency requirement and communication costs. We expect to be able to publish several papers in parallel computing, distributed systems and operating systems conferences and journals – especially in with respect to efficient scheduling strategies and resource utilization.

WP5: utilization of heterogeneous resources. Prototyping a version of the run-time capable of utilizing heterogeneous resources in a hardware resource topology will be started when the first homogeneous prototype is close to finished in WP3. Primarily, we will target the interaction between multi-core CPUs and GPUs. As such, a run-time for executing on each hardware architecture is required where similar communication requirements must be taken into account as in WP4. This also implies that the programming language will allow the specification of the architectures that is best suited for executing a given task or component. New high and low-level schedulers must be implemented to ensure that the scheduling of tasks is optimal.

Tasks	**T5.1** Implement run-time system for selected heterogeneous architecture **T5.2** Extend high and low-level scheduler for utilization of heterogeneous resources
Expected results	As in WP3, a prototype of a run-time will be developed but with support for heterogeneous devices (GPUs). We expect to be able to publish several papers in parallel computing, distributed systems and operating systems conferences and journals.

4.2 Budget

As the budget in the application form shows, the *EONS* project mainly aims to fund personnel, i.e., one postdoc and two PhD students. Additionally, there is a post in the budget for travels – to meetings, conferences and workshops and to have visits from our international collaboration partners. With respect to equipment, we will use existing infrastructure at SRL, UiO and UiT (and our collaboration partners) for experiments. All partners operate computer clusters that

will be available, and a distributed lab environment between SRL, UiO and UiT was created in the context of the iAD CRI.

5 Project Management, Organization and Collaboration

5.1 Project Management

The project will be managed by a Project Manager, an Advisory Board and an Annual Workshop. The **Project Manager** is responsible for the overall management and progress of the project. A primary objective is to coordinate the consistency of the technical work to enable integration of results, ensure that a flow of information between participants is achieved and maintain a general overview of the status of the project and its participants. The Project Manager is also responsible for the progress of the PhD students. The main task of the **Advisory Board** is to ensure the relevance of the research. The Advisory Board shall have a half to full day meeting once a year to give advice on research directions and provide input on industrial relevance. The members of this board will be senior researchers from the participating institutions, external collaborators and companies. The **Annual Workshop** will provide an opportunity to gather the project participants for dissemination of knowledge, technical discussions, progress reports and so forth. A major issue will be to review the technical results and control whether tasks in work packages are reachable.

5.2 Project Participants

Simula Research Laboratory (SRL) is represented by the Media Performance Group (MPG) whose members are also associated with **the University of Oslo** (Department of Informatics, Networks and Distributed Systems). MPG is a research group that investigates improvements to resource utilization and performance in distributed systems. Members of MPG have considerable experience with the development of software for increasing performance, especially for multimedia applications, e.g., by building run-times, enhancing operating systems, improving algorithms, offloading to GPUs, reducing resource requirements, etc. The group takes a systems approach, whereby successful research leads to quantifiable improvements and success is proven experimentally. The iAD CRI, led by MDCN/FAST and sponsored by the Norwegian Research Council, is represented by MPG members (the project proposers) at the University of Oslo. (see also http://simula.no/department/media/)

The Information Access Group, University of Tromsø focuses on fundamental structures and concepts for run-time systems supporting large scale information access applications, and builds and evaluates run-time systems for cloud computing environments. This research group is heavily involved in iAD, and has broad experience in the area of event-driven architectures supporting a more general publish/subscribe paradigm (the WAIF project). Members of the group are all employed at the department of computer science. The group has also strong international contacts in the proposed research area which they bring into the proposed project, i.e., their three adjunct positions with leading international experts in the area: Professor Fred B. Schneider (Cornell University), Professor Johannes Gehrke (Cornell University) and Dr. Cathal Gurrin (Dublin City University) (see also http://site.uit.no/iad/).

5.3 Collaboration

The project partners have for a long time collaborated in the area of the proposal, and we have earlier mentioned the collaboration in the iAD CRI including large companies with "big-data" workloads. In particular, MDCN has large activities in the area of big-data led by the Tromsø office with a very tight collaboration with the University. In addition, we have collaboration with several industrial and academic partners relevant for the proposal, both with respect to research, equipment and real-world workloads. One partner is **the Parallel and Distributed**

Processing Laboratory, National University of Defense Technology, which is a leading research institute in China on high performance computing, parallel processing and computer architecture. Tianhe-1A, the world's fastest supercomputer ranked in HPC 2010, with a peak computing rate of 2.507 petaFLOPS, has been developed and operated by this laboratory. **Dolphin Interconnect Solutions** is a Norwegian company specializing in creating high performance computing platforms for demanding applications. Dolphin has extensive experience in developing leading and innovative computer interconnect technology. Dolphin has supplied its interconnect products to major industry players including Siemens, Philips, Sony, Sun MicroSystems, Lexmark, Boeing and Thales. **The Human Media Archives Research Group, Dublin City University** (also iAD) develop and evaluate high precision search and recommendation solutions that support effective multi-modal access to multimedia content. Their research focuses on managing human media archives, specifically information retrieval, HCIR, multimedia content analysis and e-memory gathering and searching.

5.4 Risk Assessment

The *EONS* project has set ambitious goals, and as such there are associated risks. For example, the goal of implementing a proof-of-concept prototype that optimizes both a developers work process and utilization of available heterogeneous resources is a broad goal. Even though there is broad experience in the consortium, there is a possibility that certain work packages become too demanding. To accommodate for this, we have set up the work packages in a way where it is possible to down-grade the activity of some of them without being detrimental to the research efforts of the project. Moreover, a significant research contribution lies in the development of high and low-level schedulers. There is a risk that executing complex scheduling decisions introduces significant overhead when the system is reconfiguring; rendering such mechanisms extraneous. This does not mean that the research effort is without value, but might require a change in strategy for making scheduling decisions. The focus might then have to switch to offline profiling and more static schedules for disseminating tasks in the distributed system.

6 Perspectives and compliance with strategic documents

Compliance with strategic documents The *EONS* project matches the strategies of both SRL and UiT. The MPG department (SRL) aims at an integrated understanding of all aspects of distributed (multimedia) systems where an important area is a system's efficiency of processing, storage and communication. The main research areas of the Information Access Group (UiT) are large distributed systems, cluster and cloud technology. As such, the *EONS* project explores basic research questions that are shared by all the partners.

Relevance and benefit to society There exists a large number of industries, governments and institutes that need big-data processing, i.e., beyond what is available today with modern grid, cloud and cluster technologies. This area is the next frontier for innovation [20] where, for example, the Obama administration plans large initiatives in the area of distributed processing [5, 4]. Furthermore, in the future, the EU commission will be focusing on the potential of "big-data", the increasingly large and complex datasets that permeate the information economy [1, 2]. Thus, the proposed research on and development of efficient processing concepts for big-data type of workloads on future elastic heterogeneous resources may have impact and be beneficial to both national and international players where large data sets need processing in real-time.

Environmental impact No negative environmental impacts are foreseen as a result of this research. The mechanisms researched, however, will be capable of adjusting the consumption of resources based on the requirements of the workload. As such, it will not use resources it

does not require at a given time. Likewise, schedulers can be optimized to minimize energy consumption.

Ethical perspectives To the best of our knowledge, there are no ethical questions raised by the proposed project.

Gender issues The proposed *EONS* project does not have any relevant gender perspectives, but we aim to achieve a gender balance. In this respect, we will adhere to the European Commissions Code of Best Practices for Women in ICT issued in September 2009.

References

[1] Big Data–The Digital Agenda for Europe and Challenges for 2012. http://www.youtube.com/watch?v=C_8UkMHUJcQ. 1, 6

[2] Leveraging Big Data for Economic Development in the EU: what policies can the Digital Agenda propose? http://daa.ec.europa.eu/content/leveraging-big-data-economic-development-eu-what-policies-can-digital-agenda-propose. 1, 6

[3] New Activities of the Federal Big Data Initiative. http://www.cccblog.org/2013/05/02/new-activities-of-the-federal-big-data-initiative/. 1

[4] Obama administration unveils "big data" initiative: Announces $200 million in new r&d investments. http://www.whitehouse.gov/sites/default/files/microsites/ostp/big_data_press_release.pdf. 1, 6

[5] Obama's big data plans: Lots of cash and lots of open data. http://gigaom.com/cloud/obamas-big-data-plans-lots-of-cash-and-lots-of-open-data/. 1, 6

[6] Acar, U. A., Blelloch, G. E., and Blumofe, R. D. The data locality of work stealing. In *Proceedings of the twelfth annual ACM symposium on Parallel algorithms and architectures* (New York, NY, USA, 2000), SPAA '00, ACM, pp. 1–12. 2

[7] Arora, N. S., Blumofe, R. D., and Plaxton, C. G. Thread scheduling for multiprogrammed multiprocessors. In *Proceedings of ACM symposium on Parallel algorithms and architectures (SPAA)* (1998), pp. 119–129. 2

[8] Chamberlain, B. L., Callahan, D., and Zima, H. P. Parallel programmability and the Chapel language. *International Journal of High Performance Computing Applications 23*, 3 (2007). 2

[9] chih Yang, H., Dasdan, A., Hsiao, R.-L., and Parker, D. S. Map-reduce-merge: simplified relational data processing on large clusters. In *Proc. of ACM SIGMOD* (New York, NY, USA, 2007), ACM, pp. 1029–1040. 2

[10] de Kruijf, M., and Sankaralingam, K. MapReduce for the Cell BE architecture. *University of Wisconsin Computer Sciences Technical Report CS-TR-2007 1625* (2007). 2

[11] Dean, J., and Ghemawat, S. Mapreduce: simplified data processing on large clusters. In *Proc. of USENIX OSDI* (2004), pp. 10–10. 2

[12] Espeland, H., Beskow, P. B., Stensland, H. K., Olsen, P. N., Kristoffersen, S., Griwodz, C., and Halvorsen, P. P2G: A framework for distributed real-time processing of multimedia data. In *Proceedings of the International Conference on Parallel Processing (ICPP) Workshops - the International Workshop on Scheduling and Resource Management for Parallel and Distributed Systems (SRMPDS)* (Washington, DC, USA, 2011), pp. 416–426. 2

[13] Espeland, H., Olsen, P., Griwodz, C., and Halvorsen, P. Low-level scheduling implications for data-intensive cyclic workloads on modern microarchitectures. In *Proceedings of the International Workshop on Scheduling and Resource Management for Parallel and Distributed Systems* (Sept. 2012). 2

[14] Feo, J., Cann, D., and Oldehoeft, R. A report on the sisal language project* 1. *Journal of Parallel and Distributed Computing 10*, 4 (1990), 349–366. 2

[15] Gedik, B., Andrade, H., Wu, K.-L., Yu, P. S., and Doo, M. Spade: the system s declarative stream processing engine. In *Proceedings of the ACM international conference on Management of data (SIGMOD)* (2008), pp. 1123–1134. 2

[16] Halvorsen, P., Sægrov, S., Mortensen, A., Kristensen, D. K., Eichhorn, A., Stenhaug, M., Dahl, S., Stensland, H. K., Gaddam, V. R., Griwodz, C., and Johansen, D. Bagadus: An integrated system for arena sports analytics a soccer case study. In *Proceedings of the International Conference on Multimedia Systems (MMSys)* (Mar. 2013), pp. 48–59. 2, 1

[17] He, B., Fang, W., Luo, Q., Govindaraju, N. K., and Wang, T. Mars: a MapReduce framework on graphics processors. In *Proc. of PACT* (New York, NY, USA, 2008), ACM, pp. 260–269. 2

[18] Isard, M., Budiu, M., Yu, Y., Birrell, A., and Fetterly, D. Dryad: distributed data-parallel programs from sequential building blocks. In *Proc. of ACM EuroSys* (New York, NY, USA, 2007), ACM, pp. 59–72. 2

[19] Kvalnes, Å., Johansen, D., Valvåg, S., van Renesse, R., and Schneider, F. Design Principles for Isolation Kernels. Technical report 2011-70, University of Troms, 2011. 2

[20] McKinsey Global Institute. Big data: The next frontier for innovation, competition, and productivity. http://www.mckinsey.com/Insights/MGI/Research/Technology_and_Innovation/Big_data_The_next_frontier_for_innovation. 1, 6

[21] Olston, C., Reed, B., Srivastava, U., Kumar, R., and Tomkins, A. Pig latin: a not-so-foreign language for data processing. In *Proc. of ACM SIGMOD* (New York, NY, USA, 2008), ACM, pp. 1099–1110. 2

[22] Pike, R., Dorward, S., Griesemer, R., and Quinlan, S. Interpreting the data: Parallel analysis with Sawzall. *Sci. Program. 13*, 4 (2005), 277–298. 2

[23] Ranger, C., Raghuraman, R., Penmetsa, A., Bradski, G., and Kozyrakis, C. Evaluating MapReduce for multi-core and multiprocessor systems. In *Proc. of IEEE HPCA* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 13–24. 2

[24] Valvåg, S. V., and Johansen, D. Oivos: Simple and efficient distributed data processing. In *Proceeedings of IEEE International Conference on High Performance Computing and Communications (HPCC)* (2008), pp. 113–122. 2

[25] Valvåg, S. V., and Johansen, D. Cogset: A unified engine for reliable storage and parallel processing. In *Proceedings of IFIP International Conference on Network and Parallel Computing Workshops (NPC)* (2009), pp. 174–181. 2

[26] Vrba, Ž., Halvorsen, P., and Griwodz, C. A simple improvement of the work-stealing scheduling algorithm. In *Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems (CISIS) - International Workshop on Multi-Core Computing Systems (MuCoCoS)* (2010), pp. 925–930. 2

[27] Vrba, Ž., Halvorsen, P., Griwodz, C., Beskow, P., Espeland, H., and Johansen, D. The Nornir run-time system for parallel programs using Kahn process networks on multi-core machines - a flexible alternative to MapReduce. *Journal of Sumpercomputing 27*, 1 (2010). 2

[28] Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., and Stoica, I. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *Proceedings of the 5th European conference on Computer systems (EuroSys)* (2010), pp. 265–278. 2